International Journal of Computer Sciences and Engineering
Vol.**12**, Issue.**4**, pp.**86-100**, April **2024**
ISSN: 2347-2693 (Online)
Available online at: www.ijcseonline.org

ISSN: 2347-2693 (E)

Research Article

Integration of Machine Learning Algorithm into IDS-ATiC-AODV (Improved Data Security - Avoid Time Complexity Ad-Hoc On-Demand Distance Vector) Routing Protocol

N. Kanimozhi[1*], **S. Hari Ganesh**[2], **B. Karthikeyan**[3]

[1,2]Dept. of Computer Science, H.H The Rajah's College, Pudukkotai – 622 001. India
[3]Dept. of Computer Science, Bishop Heber College, Trichy – 620 017, India

**Corresponding Author: nkanimozhimphil@gmail.com*

Received: 05/Mar/2024; **Accepted**: 06/Apr/2024; **Published**: 30/Apr/2024. **DOI**: https://doi.org/10.26438/ijcse/v12i4.86100

Abstract: The Mobile Ad-Hoc Network presents a remarkable infrastructure-free approach for information exchange between source and destination utilizing intermediate nodes. It offers robust security features and tackles time complexity through its routing protocols. In the author's prior research, solutions were proposed for approximately eight qualitative and quantitative Quality of Service (QoS) metrics, along with a reduction in solution algorithm execution time. These solutions were automated using Machine Learning (ML) algorithms, leveraging a dataset named Infra-Less KMS and an optimal algorithm, Support Vector Machine (SVM) & Gaussian mixture model (GMM) identified in previous works. In this study, the IDS-ATiC AODV Solution algorithm will be implemented using SVM, with a focus on evaluating prediction accuracy.

Keywords: Security and Time Complexity, QoS, SVM, GMM, Prediction Accuracy.

1. Introduction

The Mobile Ad-Hoc Network (MANET) presents various security and time complexity challenges, such as Packet Delivery Ratio issues, End-to-End time delays, concerns about unreliable nodes, and potential attacks like Black Hole and Grey Hole attacks, as well as link breaks. The IDS-ATiC AODV routing protocol emerges as a comprehensive solution to address these challenges. Initially, these algorithms are implemented directly within simulators and thoroughly evaluated, demonstrating excellent performance across different scenarios. When the MANET experiences low traffic, the IDS-ATiC AODV routing protocol yields outstanding results both with and without intruders. However, as network traffic increases, the overhead and Normalized Routing Load (NRL) tend to escalate, leading to longer communication times in higher traffic scenarios.

Communication time denotes the duration taken for packets to travel from the source to the destination. The solution algorithm aims to minimize End-to-End Time Delay (EETD). However, an increase in Overhead and Normalized Routing Load can lead to a rise in the normal EETD. This, in turn, may cause a decrease in Packet Delivery Ratio (PDR). Traditionally, drops in PDR could be attributed to issues such as Black Holes, Unreliable Nodes, and poor bandwidth. However, now, drops in PDR can also occur due to the execution of the solution algorithm.

The solution algorithm comprises a collection of algorithms, each addressing specific network issues. Whenever a particular issue arises in the network and satisfies specific conditions, it triggers the execution of the solution algorithm. At such times, all algorithms may execute, or only a subset may execute depending on the specific conditions. Regardless, the execution may lead to increased Overhead and Normalized Routing Load, consequently extending the execution time.

Efforts should be made to decrease execution time. Integrating Machine Learning (ML) with the solution algorithm can provide a promising path forward. The author's prior research has already paved the way for incorporating ML algorithms. The creation of the Infra-Less KMS dataset was specifically tailored for integrating ML techniques, while the author's recent work identified the optimal ML algorithm, Support Vector Machine (SVM) and Gaussian Mixture Model (GMM), for enhancing the IDS-ATiC AODV routing protocol.
This work tells how to utilize SVM & GMM ML algorithm with IDS-ATiC AODV routing algorithm by the use of Infra-Less KMS dataset.

2. Review of Literature

In their study, Jhansi Rani Kaka et al. [10] introduced the DE-MSVM model aimed at improving the classification performance of Alzheimer's disease. They utilized ADNI fMRI and PET images for testing the effectiveness of Alzheimer's classification. A normalization method was employed to enhance the quality of the images. Feature extraction using the AlexNet method and feature selection through DE were conducted to identify relevant features for classification. Subsequently, the MSVM model was applied to the selected features to classify Alzheimer's images. This section provides a comprehensive overview of the results obtained from the proposed DE-MSVM approach.

Sherif Abdelfattah et al. [11] present a methodology where the term "hospital" (H) refers to the entity owning the model, while "patient" (P) represents the user of the model. The proposed approach unfolds in four distinct phases. Firstly, during the system initialization stage, the key distribution center (KDC) computes and distributes secret keys to both H and P. Moving on to the model encryption phase, each parameter vector of the support vector machine (SVM) model is encrypted by H. These encrypted parameters, along with random numbers used for masking classification results, are then outsourced to the cloud server (CS). In the subsequent step of medical data encryption, P encrypts their medical data vector, including symptoms and vital data, before transmitting it to the server for input to the diagnosis model and computation of the masked classification score in the medical diagnosis phase. Finally, these masked classifications are sent back to P for unmasking and subsequent interpretation.

Ceren Atik et al. [12] introduce a novel approach called Support Vector Machine Chains (SVMC), which employs a structured method where a series of models are trained with attribute reduction at each stage. Predictions from each learner within the chain are combined using a unique voting mechanism known as tournament voting. This voting process involves dividing classifier results by the tournament size and subsequently employing a selection approach based on class labels within groups for further processing. The approach iteratively progresses through subsequent rounds until the end of the tournament, where the winning class label is designated as the final prediction.

Zhi Quan et al. [13] emphasize the significant impact Support Vector Machines (SVM) have made in various fields such as modern machining, protein prediction, and face detection. They highlight the convenience of SVM's reliance solely on support vectors for model determination, facilitating training processes. However, they also acknowledge drawbacks, particularly the inefficiency of existing SVM models in training large-scale datasets for practical applications. Thus, future SVM development may focus on enhancing algorithmic efficiency. Furthermore, while SVM exhibits theoretical advantages, its practical application research lags behind. Therefore, future research is poised to concentrate on expanding SVM's applicability in everyday life and exploring novel application domains.

Siva Rajesh Kasa et al. [14] address the issue of subpar solutions arising from clustering with improperly specified Gaussian Mixture Models (GMMs). These solutions exhibit distinct characteristics, including asymmetrical component orientation and sizes, as well as varying frequency of occurrence compared to spurious solutions. Through theoretical analysis, we unveil a novel relationship between the asymmetry of fitted components and model misspecification. Future exploration of this correlation promises to be intriguing.

Abdullahi Abubekar Mas'ud et al. [15] employ Gaussian Mixture Models (GMM) for clustering and classifying electrical trees emerging from epoxy resin insulation in this study. Various partial discharge (PD) samples are collected at different voltages spanning from initial to final breakdown stages. Findings reveal that PD dynamics vary depending on stressing voltages and tree growth levels, leading to different breakdown times. GMM is favored over alternative methods due to its robustness and capability to perform hard clustering on intricate data like electrical tree patterns. Results demonstrate GMM's effectiveness in classifying patterns from initial to breakdown levels for breakdown times exceeding an hour, though not for times under an hour, particularly with samples stressed at the highest voltage of 16 kV. PD patterns for shorter breakdown times exhibit similar clusters across degradation stages. Cluster centers and confidence intervals are developed to recognize PD patterns at various stages. Nonetheless, further validation through experimentation with different samples at varied voltages and breakdown times is warranted. Additionally, exploring different insulating materials such as polyethylene or cross-linked polyethylene could further evaluate the proposed classification tool's efficacy.

3. Support Vector Machine (SVM)

SVM stands for Support Vector Machine, a supervised machine learning technique applicable to both classification and regression problems.

Its primary function is to establish an optimal boundary, often referred to as a hyperplane, between distinct classes within a dataset. In essence, SVM undertakes intricate data transformations based on a selected kernel function, with the objective of maximizing the margin between data points.

• SVM Operation: In its basic form, particularly in cases of linear separability, SVM endeavours to locate a line that maximizes the margin between two classes within a two-dimensional space.

• SVM Objective: The main aim of SVM is to identify a hyperplane that best separates data points according to their respective classes within an n-dimensional space. The data points situated closest to this hyperplane, known as support vectors, play a crucial role in defining the separation.

• Illustration: As depicted in the provided diagram, the support vectors are evident as three points aligned along scattered lines, encompassing two blue and one green support vectors. The separation hyperplane is visually represented by the solid red line.

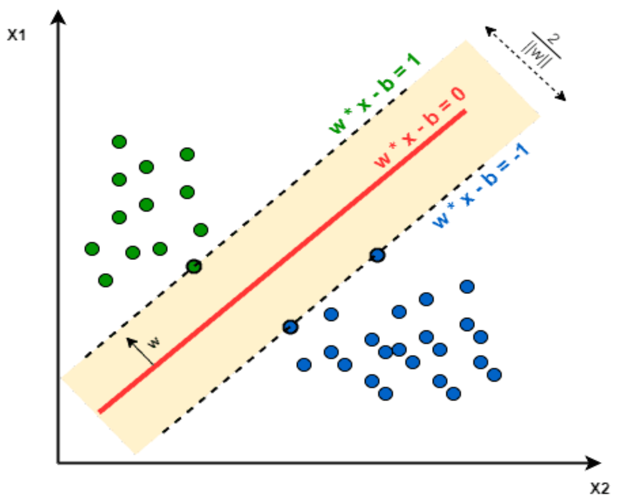


Figure 1: SVM: Three point aligned along scattered lines.

3.1. Multiclass Classification Using SVM

In its most fundamental form, SVM lacks native support for multiclass classification. To address this, multiclass problems are typically decomposed into multiple binary classification tasks, allowing SVM to be applied.
Various methods are commonly employed for multiclass classification using SVM (multiclass support vector machines), including:

- One vs One (OVO) approach
- Directed Acyclic Graph (DAG) approach
- One vs All (OVA) approach

Now, let's delve into each of these strategies individually, examining their intricacies

3.1.1. One vs One (OVO)

This strategy breaks down our multiclass classification task into binary subproblems. As a result, we generate binary classifiers for each pair of classes. Final predictions are made by combining majority voting with the confidence criterion based on distance from the margin.
However, a notable drawback of this method is the need to train multiple SVMs.
In the context of multi-class/multi-label problems with L categories, let's consider the (s, t)-th classifier:

- Positive Samples: all points in class s ({ xi: s ∈ yi })
- Negative Samples: all points in class t ({ xi: t ∈ yi })
- fs, t(x): decision value of this classifier (where a larger value of fs, t(x) indicates a higher probability for label s over label t)
- f t, s(x) = – f s, t(x)
- Prediction: f(x) = argmax s (Σ t fs, t(x))

For example, let's envision a scenario involving three-class classification problems: Green, Red, and Blue.

In the One-to-One approach, the objective is to find the hyperplane that effectively separates each pair of classes while disregarding the presence of points from the third class. For instance, the Red-Blue line focuses solely on maximizing the distinction between blue and red points, without factoring in the presence of green points

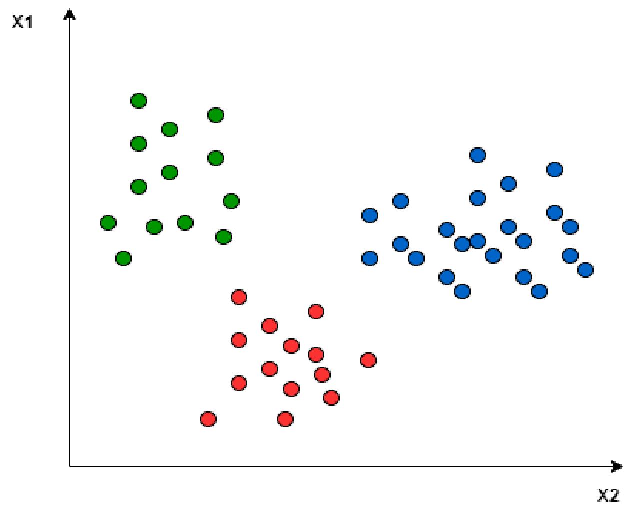


Figure 2: SVM-Three-Class Classification

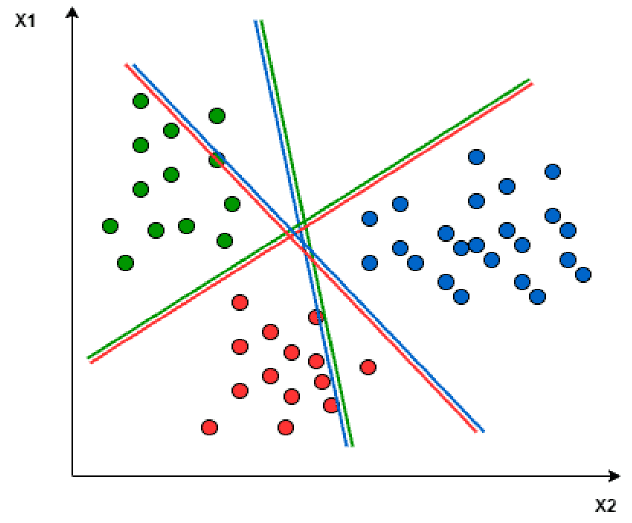


Figure 3: Multiple classification

3.1.2. One vs All (OVA)

In this method, for an N-class problem, we train N SVMs:
SVM number -1 learns “class_output = 1” versus “class_output ≠ 1”
SVM number -2 learns “class_output = 2” versus “class_output ≠ 2”
…
…
SVM number -N learns “class_output = N” versus “class_output ≠ N”
To predict the output for new input data, we predict with each of the trained SVMs and then determine which one yields the prediction farthest into the positive region, serving as a confidence criterion for a specific SVM.

3.1.3. Challenges with N SVM’s

However, training these N SVMs poses challenges:
High Computational Complexity: Implementing the One vs All (OVA) strategy requires processing more training points, leading to increased computational demands.

Imbalanced Data: For instance, in an MNIST dataset with 10 classes (0 to 9) and 1000 points per class, one SVM may be trained on 9000 points while another has only 1000, resulting in an imbalance.
To mitigate this issue:

- Utilize the 3-sigma rule of the normal distribution to fit data and subsample accordingly, maintaining class distribution.
- Randomly select data points from the majority class.
- Employ popular subsampling techniques like SMOTE.

For multi-class/multi-label problems with L categories:
For the t-th classifier:

- Positive Samples: all points in class t ({ xi: t ∈ yi })
- Negative Samples: all points not in class t ({ xi: t ∉ yi })
- ft(x): the decision value for the t-th classifier (where a larger value of ft indicates a higher probability that x is in class t)
- Prediction: f(x) = argmax t ft(x)

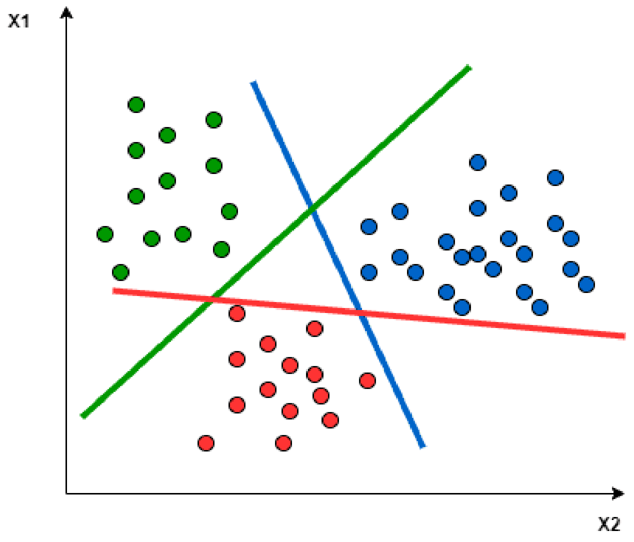


Figure 4: One vs All approach

In the **One vs All** approach, a hyperplane is sought to separate the classes, considering all points and dividing them into two groups: one for the points of the class under analysis and another for all other points
A single SVM performs binary classification, distinguishing between two classes as follows:
• The One vs All approach utilizes L SVMs for classification.
• The One vs One approach employs L(L-1)/2 SVMs for classification

4. Gaussian Mixture Model (GMM)

The Gaussian mixture model (GMM) represents a set of distributions over vectors with real values in
$\mathbb{R}^n$. The definition of the GMM is as follows: Initially, we posit the existence of K Gaussian distributions. Next, to generate a sample $\boldsymbol{x} \in \mathbb{R}^n$, we begin by choosing one of these K Gaussians based on a Categorical distribution

$$Z \sim \text{Cat}(\alpha_1, \dots, \alpha_K) \qquad \text{……01}$$

Here, Z∈{1,2,…,K} indicates which Gaussian to select (for instance, if Z=2, we pick the 2nd Gaussian), and α_k represents the probability of selecting the k th Gaussian. Once Z=z, we then sample X from that z th Gaussian. In other words,

$$X \sim N(\mu_z, \Sigma_z) \qquad \text{……02}$$

Here, μ_z denotes the mean of the z th Gaussian, and Σ_z represents its covariance matrix.
This model is represented by the following graphical structure:

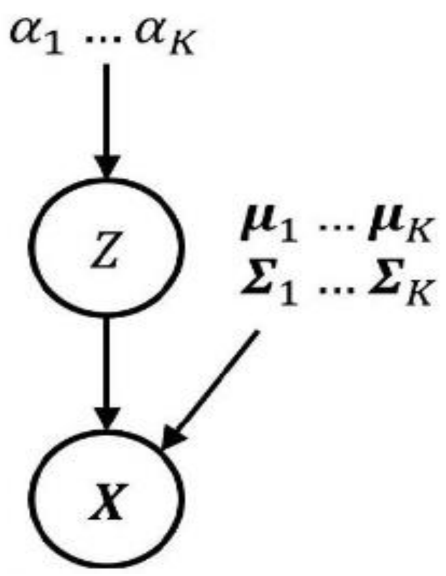


Figure 5: Gaussian Mixture Model

Each of the K Gaussian distributions possesses its distinct set of parameters, including a mean vector and a covariance matrix.
Additionally, the probabilities α_1,…,α_k associated with selecting the Gaussians are also considered as parameters of the model. We will refer to this comprehensive collection of model parameters as

$$\Theta = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K, \boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K, \alpha_1, \dots, \alpha_K\} \qquad \text{……03}$$

In scenarios where GMMs are employed, such as in the context of data clustering (which will be further explored), Z is commonly unobserved. Consequently, our attention is directed towards the marginal distribution of X, characterized by its density function:

$$p(\boldsymbol{x}; \Theta) := \sum_{k=1}^{K} P(Z = k; \Theta) p(\boldsymbol{x} \mid Z = k; \Theta)$$
$$= \sum_{k=1}^{K} \alpha_k \phi(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad \text{……04}$$

where ϕ represents the probability density function of the multivariate Gaussian distribution:

$$\phi(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) := \frac{1}{(2\pi)^{\frac{n}{2}} \det(\boldsymbol{\Sigma})^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right] \qquad \text{……05}$$

Below is an illustrative density function for a two-dimensional GMM featuring three Gaussians (i.e., K=3)

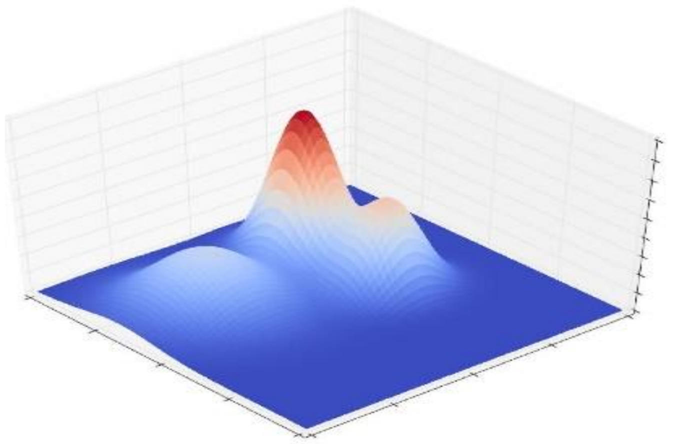


Figure 6: Two-dimensional GMM

4.1. A framework for clustering data

Let's consider a dataset containing points$\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n \in \mathbb{R}^n$, and our objective is to identify clusters among these data points where points within a cluster exhibit greater similarity to each other compared to points outside their cluster. GMMs offer a framework for uncovering such clusters.

To cluster the data, we start with a strong assumption: that our data points were generated from a GMM with K Gaussians (where K represents the assumed number of clusters). However, we lack knowledge about the GMM's parameters, including the means and covariances of each Gaussian, as well as the assignment of each data point to a specific Gaussian (represented by the random variables $Z_1, \dots, Z_n$).

This scenario is depicted in the illustration below. On the left side, we have an ideal scenario where we know the parameters of the GMM and have samples $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$generated from that model. Importantly, we also have information about which Gaussian generated each data point - denoted as $z_1, \dots, z_n$ (these are indicated by the colors of the data points). On the right side, we are only provided with $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$, lacking knowledge about the model parameters Θ and the assignment of data points to Gaussians, $z_1, \dots, z_n$. In other words, $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$constitutes the observed data, while $z_1, \dots, z_n$represents the latent data.

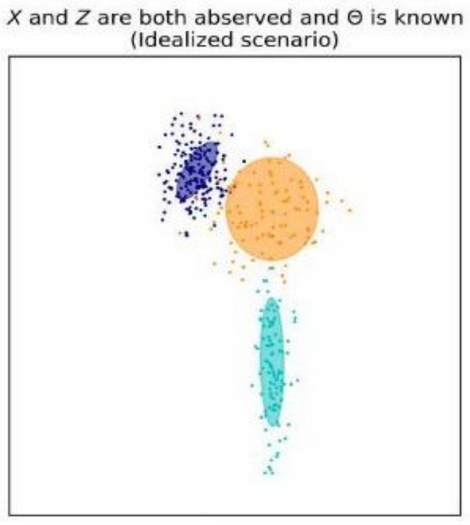


Figure 7: Idealized scenario of GMM

Only X is observable (Realistic scenario)

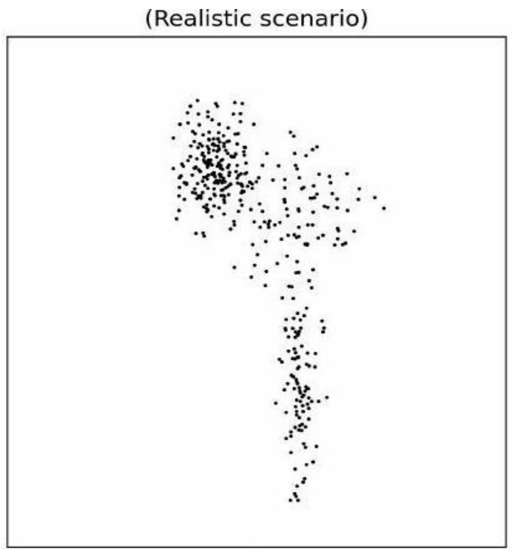


Figure 8: Realistic scenario of GMM

To conduct clustering on $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$, we can proceed with the following steps. Initially, we must estimate the values for Θ. This estimation process will be extensively discussed in this blog post, but for now, let's assume we have an estimate denoted as

$$\hat{\Theta} := \{\hat{\boldsymbol{\mu}}_1, \dots, \hat{\boldsymbol{\mu}}_K, \hat{\boldsymbol{\Sigma}}_1, \dots, \hat{\boldsymbol{\Sigma}}_K, \hat{\alpha}_1, \dots, \hat{\alpha}_K\} \quad \dots\dots 06$$

With this estimate in hand, we can allocate $\boldsymbol{x}_i$to the Gaussian (or cluster) that is deemed most probable to have generated $\boldsymbol{x}_i$:

$$\arg\max_{k \in \{1,\dots,K\}} P(Z_i = k \mid \boldsymbol{x}_i; \hat{\Theta}) \quad \dots\dots 07$$

Applying Bayes' rule, we obtain

$$P(Z_i = k \mid \boldsymbol{x}_i; \hat{\Theta}) = \frac{p(\boldsymbol{x}_i \mid Z_i = k; \hat{\Theta}) P(Z_i = k; \hat{\Theta})}{\sum_{h=1}^{K} p(\boldsymbol{x}_i \mid Z_i = h; \hat{\Theta}) P(Z_i = h; \hat{\Theta})} = \frac{\phi(\boldsymbol{x}_i \mid \hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k) \hat{\alpha}_k}{\sum_{h=1}^{K} \phi(\boldsymbol{x}_i \mid \hat{\boldsymbol{\mu}}_h, \hat{\boldsymbol{\Sigma}}_h) \hat{\alpha}_h} \quad \dots\dots 08$$

This process is illustrated in the figure below. On the left-hand side, we represent our estimation for Θ. On the right-hand side, we allocate each point to the Gaussian that is determined to be the most probable origin for that point

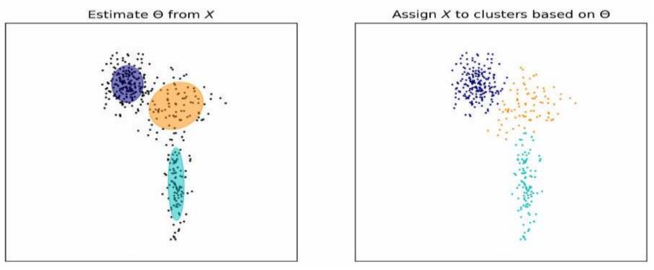


Figure 9: Identification of Cluster

Notice that we have identified three clusters in the data corresponding to the three Gaussians.

Now, let's return to the objective of estimating values for Θ. This can be pursued using the principle of maximum likelihood:

$$\hat{\Theta} := \arg\max_{\Theta} \prod_{i=1}^{n} p(\boldsymbol{x}_i; \Theta) \quad \dots\dots 09$$

How can we address this optimization problem? It happens that the EM algorithm offers a straightforward approach, as we'll explore in detail later in this post

4.2. GMM for Cluster

When using GMMs for clustering, there are several key considerations to bear in mind. Firstly, GMMs necessitate the user to specify K, which represents the number of Gaussians posited to have generated the data.

This K value determines the number of clusters sought by the algorithm. If K is set too low, the maximum-likelihood estimation of the model may result in grouping Gaussians that encompass multiple 'true' clusters. Conversely, if K is set too high, some 'true' clusters might be fragmented into multiple smaller clusters.
For instance, consider the dataset below, which was generated with three Gaussians (left), but we are fitting a GMM with two Gaussians (middle) and five Gaussians (right):

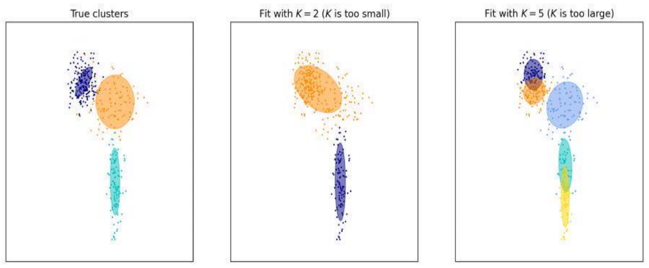


Figure 9: Colour based identification of Cluster

On the left, each point is coloured based on the Gaussian from which it was sampled. It's evident that when K is too small (middle), two of the true clusters are merged into a single cluster. Similarly, when K is too large (right), some of the true clusters are fragmented into multiple smaller clusters.

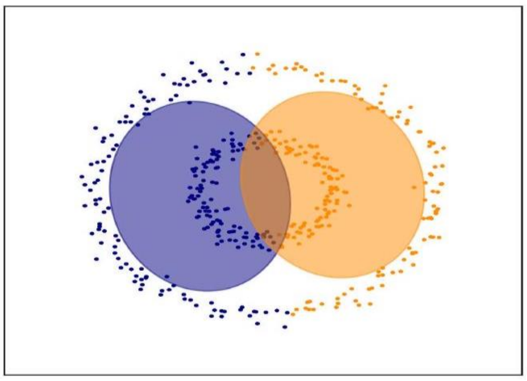


Figure 10: Identifying cluster shape

Another crucial aspect is that GMMs excel at identifying clusters shaped like 'blobs'—sets of data points forming ellipsoid-like clusters. However, when the data exhibits more intricate structures, GMMs may not yield clusters that align with intuitive interpretations. For instance, consider a dataset comprising two concentric rings. A GMM seeking two 'blob-like' clusters will struggle to separate the inner ring from the outer ring. As depicted below, the GMM divides each ring in half and assigns each half to a cluster:

4.3. Estimating maximum likelihood for GMMs using the EM algorithm

The EM algorithm is a straightforward option for conducting maximum likelihood estimation of GMM parameters due to its simplicity in implementation. For a comprehensive examination of the EM algorithm, please refer to my earlier blog post.
The EM algorithm necessitates iteratively alternating between an E-step and an M-step until the parameters converge

4.3.1. E-Step

In the E-step, we need to define the Q-function. Let's denote a given iteration of the algorithm as t. The Q-function for the t th iteration is formulated as follows:

$$Q_t(\Theta) := \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{t,i,k} \log[\alpha_k \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)] \qquad \ldots\ldots 10$$

Where

$$\gamma_{t,i,k} := \frac{\alpha_{t,k}\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_{t,k}, \boldsymbol{\Sigma}_{t,k})}{\sum_{h=1}^{K} \alpha_{t,h}\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_{t,h}, \boldsymbol{\Sigma}_{t,h})} \qquad \ldots\ldots 11$$

and $\alpha_{t,k}, \boldsymbol{\mu}_{t,k}$, and $\boldsymbol{\Sigma}_{t,k}$ are the t th estimates of $\alpha_k, \boldsymbol{\mu}_k$, and $\boldsymbol{\Sigma}_k$ respectively.

It's worth mentioning that the primary computation required at this step involves calculating the $\gamma_{t,i,l}$variables. As demonstrated in the derivation, these variables indicate the likelihood that each data point was sampled from each Gaussian, as estimated using the parameter set Θ_t. In essence, $\gamma_{t,i,l}$signifies the probability that data point i was generated by the k th Gaussian, as estimated at the t th iteration of the algorithm.

4.3.2. M-Step

During the M-step, our objective is to determine Θ that maximizes Q_t (Θ). In other words, we need to calculate the following:

$$\Theta_{t+1} := \arg\max_{\Theta} Q_t(\Theta) \qquad \ldots\ldots 12$$

The solution to this optimization problem is expressed as :

$$\forall k, \alpha_{t+1,k} := \frac{1}{n} \sum_{i=1}^{n} \gamma_{t,i,k}$$

$$\forall k, \boldsymbol{\mu}_{t+1,k} := \frac{1}{\sum_{i=1}^{n} \gamma_{t,i,k}} \sum_{i=1}^{n} \gamma_{t,i,k} \boldsymbol{x}_i \qquad 13$$

$$\forall k, \boldsymbol{\Sigma}_{t+1,k} := \frac{1}{\sum_{i=1}^{n} \gamma_{t,i,k}} \sum_{i=1}^{n} \gamma_{t,i,k} (\boldsymbol{x}_i - \boldsymbol{\mu}_{t,k})(\boldsymbol{x}_i - \boldsymbol{\mu}_{t,k})^T$$

4.3.3. Hot Steps

The following code snippet describes the EM algorithm:

While
$(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n; \Theta_t) - p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n; \Theta_{t-1}) < \epsilon$ 14

$$\forall k, \forall i, \gamma_{t,i,k} \leftarrow \frac{\alpha_{t,k}\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_{t,k}, \boldsymbol{\Sigma}_{t,k})}{\sum_{h=1}^{K} \alpha_{t,h}\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_{t,h}, \boldsymbol{\Sigma}_{t,h})}$$

$$\forall k, \alpha_{t+1,k} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \gamma_{t,i,k}$$

$$\forall k, \boldsymbol{\mu}_{t+1,k} \leftarrow \frac{1}{\sum_{i=1}^{n} \gamma_{t,i,k}} \sum_{i=1}^{n} \gamma_{t,i,k}\boldsymbol{x}_i$$

$$\forall k, \boldsymbol{\Sigma}_{t+1,k} \leftarrow \frac{1}{\sum_{i=1}^{n} \gamma_{t,i,k}} \sum_{i=1}^{n} \gamma_{t,i,k}(\boldsymbol{x}_i - \boldsymbol{\mu}_{t,k})(\boldsymbol{x}_i - \boldsymbol{\mu}_{t,k})^T$$

$$t \leftarrow t+1$$

ϵ is a parameter provided by the user that governs convergence criteria. Additionally, it's important to note that the initial parameters $\Theta_0$are not critical and can be arbitrarily set. Since the EM algorithm converges to a local maximum, it's often beneficial to run EM multiple times with different initial values of $\Theta_0$and select the solution that maximizes the likelihood function.

4.3.4. The EM algorithm in operation

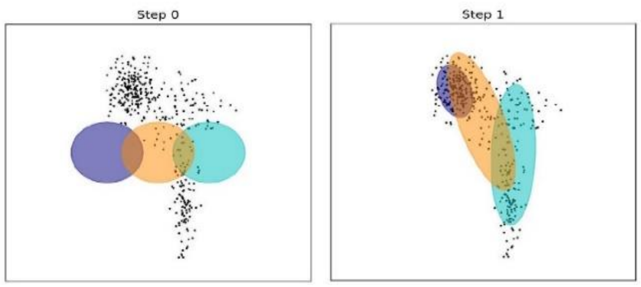


Figure 11: EM algorithm for GMM

In the illustration above and below, we showcase the EM algorithm for GMMs in progress. As the number of iterations increases, the means and covariances of each Gaussian gradually converge, aligning the Gaussians with the three clusters in the data:

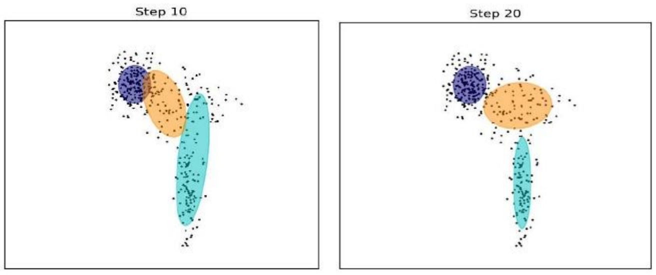


Figure 11 Gaussians with the three clusters

4.3.5. Developing the EM algorithm for Gaussian Mixture Models

Derivation of the E-step

Here, we establish the Q-function at the t th iteration. Let X:={ $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$} denote the set of observed data (i.e., the data points), and Z:={ $z_1, \dots, z_n$} represent the collection of latent data (i.e., indicating which Gaussian generated each data point). As a reminder, the Q-function is defined as:

$$Q_t(\Theta) := E_{Z|X;\Theta_t}[\log p(X, Z; \Theta)] \quad \text{……15}$$

Obtaining the Q-function involves deriving an analytical expression for this expectation, enabling its implementation in a computer program. For GMMs, this derivation is as follows

$$\begin{aligned} Q_t(\Theta) &:= E_{Z|X;\Theta_t}[\log p(X, Z; \Theta)] \\ &= \sum_{i=1}^{n} E_{z_i|x_i;\Theta_t}[\log p(\boldsymbol{x}_i, z_i; \Theta)] \text{ by the linearity of ex} \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} P(z_i = k \mid \boldsymbol{x}_i; \Theta_t)\log p(\boldsymbol{x}_i, z_i; \Theta) \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \frac{P(\boldsymbol{x}_i \mid z_i = k; \Theta_t)P(z_i = k; \Theta_t)}{\sum_{h=1}^{K} P(\boldsymbol{x}_i \mid z_i = h; \Theta_t)P(z_i = h; \Theta_t)}\log \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \frac{\alpha_{t,k}\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_{t,k}, \boldsymbol{\Sigma}_{t,k})}{\sum_{h=1}^{K} \alpha_{t,h}\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_{t,h}, \boldsymbol{\Sigma}_{t,h})}\log \alpha_k\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_{k} \\ &= \sum_{i=1}^{n}\sum_{k=1}^{K} \gamma_{t,i,k}\log \alpha_k\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \end{aligned} \quad \text{……16}$$

where we let

$$\gamma_{t,i,k} := \frac{\alpha_{t,k}\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_{t,k}, \boldsymbol{\Sigma}_{t,k})}{\sum_{h=1}^{K} \alpha_{t,h}\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_{t,h}, \boldsymbol{\Sigma}_{t,h})} \quad \text{……17}$$

4.3.6. Developing the M-step

In the M-step, the objective is to determine Θ_{t+1}that maximizes the Q-function.

It's important to note that the $\alpha_1, \dots, \alpha_k$represent the probabilities of a given point being sampled from each Gaussian, and consequently, these probabilities must sum to one. Therefore, when optimizing the Q-function with respect to these parameters, it's imperative to do so under the constraint that they sum to one. In other words, we need to solve the following equation:

$$\Theta_{t+1} := \arg\max_{\Theta} Q_t(\Theta) \quad \text{……18}$$

with the condition that

$$\sum_{k=1}^{K} \alpha_k = 1 \quad \text{……19}$$

Due to the presence of the equality constraint and the continuity of both the objective and the constraint, this optimization problem can be addressed using Lagrange multipliers. Initially, we construct the Lagrangian as follows:

$$L(\Theta, \lambda) := \sum_{i=1}^{n}\sum_{k=1}^{K}\gamma_{t,i,k}\log \alpha_k\phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) + \lambda(\textstyle\sum_{k=1}^{K}\alpha_k - 1) \quad \text{……20}$$

Now, we aim to find Θ that satisfies the condition where the derivative of the Lagrangian equals zero.

We begin with $\alpha_1, \dots, \alpha_K$. For a particular k,

$$\frac{\partial L(\Theta, \lambda)}{\partial \alpha_k} = \frac{1}{\alpha_k}\sum_{i=1} \gamma_{t,i,k} + \lambda \quad \text{……21}$$

By setting it to zero and solving for α_k, we obtain:

$$\begin{aligned} 0 &= \frac{1}{\alpha_k}\sum_{i=1} \gamma_{t,i,k} + \lambda \\ \Rightarrow \alpha_k &= -\frac{1}{\lambda}\sum_{i=1}^{n} \gamma_{t,i,k} \end{aligned} \quad \text{……22}$$

Substituting this into the constraint $\sum_{k=1}^{K}\alpha_k = 1$, we can determine λ by solving

$$\sum_{k=1}^{K} -\frac{1}{\lambda}\sum_{i=1}^{n} \gamma_{t,i,k} = 1$$

$$\Rightarrow \lambda = -\sum_{i=1}^{n}\sum_{k=1}^{K} \gamma_{t,i,k} \qquad \ldots 23$$

$$\Rightarrow \lambda = -n \text{ because } \sum_{k=1}^{K} \gamma_{t,i,k} = 1$$

Lastly, substituting λ back into the equation that equates the derivative of the Lagrangian to zero, we can calculate the ultimate value of α_k:

$$0 = \frac{1}{\alpha_k}\sum_{i=1} \gamma_{t,i,k} + \lambda$$

$$\Rightarrow 0 = \frac{1}{\alpha_k}\sum_{i=1} \gamma_{t,i,k} - n \qquad \ldots 24$$

$$\Rightarrow \alpha_k = \frac{1}{n}\sum_{i=1} \gamma_{t,i,k}$$

And with that, we have determined the $\alpha_1, \ldots, \alpha_k$parameters that maximize the Q-function.

Next, we proceed to the Gaussian means. Initially, we calculate the derivative of the Lagrangian with respect to $\boldsymbol{\mu}_k$for a specific k:

$$\frac{\partial L(\Theta,\lambda)}{\partial \boldsymbol{\mu}_k} := \sum_{i=1}^{n} \gamma_{t,i,k}\frac{\partial}{\partial \boldsymbol{\mu}_k}\log \alpha_k \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$
$$= \sum_{i=1}^{n} \frac{\gamma_{t,i,k}}{\alpha_k \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}\frac{\partial}{\partial \boldsymbol{\mu}_k}\alpha_k \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \qquad 25$$
$$= \sum_{i=1}^{n} \gamma_{t,i,k}[-2\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)]$$

The final step of this derivation is based on the fact that

$$\frac{\partial}{\partial s}(\boldsymbol{x}-\boldsymbol{s})^T \boldsymbol{W}(\boldsymbol{x}-\boldsymbol{s}) = -2\boldsymbol{W}^{-1}(\boldsymbol{x}-\boldsymbol{s}) \qquad \ldots\ldots.26$$

Now, by equating the derivative of the Lagrangian with respect to the mean vector to the zero vector, we can determine $\boldsymbol{\mu}_k$:

$$\boldsymbol{0} = \sum_{i=1}^{n} \gamma_{t,i,k}[-2\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)]$$

$$\Rightarrow \boldsymbol{0} = \boldsymbol{\Sigma}_k^{-1}\sum_{i=1}^{n} \gamma_{t,i,k}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)$$

$$\Rightarrow \boldsymbol{\Sigma}_k\boldsymbol{0} = \boldsymbol{\Sigma}_k\boldsymbol{\Sigma}_k^{-1}\sum_{i=1}^{n} \gamma_{t,i,k}(\boldsymbol{x}_i - \boldsymbol{\mu}_k) \qquad \ldots\ldots 27$$

$$\Rightarrow \boldsymbol{0} = \sum_{i=1}^{n} \gamma_{t,i,k}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)$$

$$\Rightarrow \boldsymbol{\mu}_k = \frac{1}{\sum_{i=1}^{n} \gamma_{t,i,k}}\sum_{i=1}^{n} \gamma_{t,i,k}\boldsymbol{x}_i$$

And now, we have found the Gaussian means that maximize the Q-function.

Lastly, we address the determination of the covariance matrices that maximize the Q-function. Initially, we calculate the gradient with respect to the covariance matrix $\boldsymbol{\Sigma}_k$for a specific k:

$$\frac{\partial L(\Theta,\lambda)}{\partial \boldsymbol{\Sigma}_k} := \sum_{i=1}^{n} \gamma_{t,i,k}\frac{\partial}{\partial \boldsymbol{\Sigma}_k}\log \alpha_k \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$
$$= \sum_{i=1}^{n} \frac{\gamma_{t,i,k}}{\alpha_k \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}\frac{\partial}{\partial \boldsymbol{\Sigma}_k}\alpha_k \phi(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$
$$= \sum_{i=1}^{n} \gamma_{t,i,k}\left[-\frac{1}{2}\frac{\partial}{\partial \boldsymbol{\Sigma}_k}\log\det(\boldsymbol{\Sigma}_k) - \frac{1}{2}\frac{\partial}{\partial \boldsymbol{\Sigma}_k}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)\right] \qquad 28$$
$$= \sum_{i=1}^{n} \gamma_{t,i,k} - \frac{1}{2}\boldsymbol{\Sigma}_k^{-1} - \gamma_{t,i,k}\frac{1}{2}\frac{\partial}{\partial \boldsymbol{\Sigma}_k}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k) \text{ See Note 1 below}$$
$$= -\frac{1}{2}\sum_{i=1}^{n} \gamma_{t,i,k}\boldsymbol{\Sigma}_k^{-1} - \gamma_{t,i,k}\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_i)(\boldsymbol{x}_i - \boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_k^{-1} \text{ See Note 2 below}$$
$$= -\frac{1}{2}\left(\sum_{i=1}^{n} \gamma_{t,i,k}\right)\boldsymbol{\Sigma}_k^{-1} - \boldsymbol{\Sigma}_k^{-1}\left(\sum_{i=1}^{n} \gamma_{t,i,k}(\boldsymbol{x}_i - \boldsymbol{\mu}_i)(\boldsymbol{x}_i - \boldsymbol{\mu}_i)^T\right)\boldsymbol{\Sigma}_k^{-1}$$

5. IDS-ATiC AODV routing algorithm

The diagram Figure 12 illustrates the IDS-ATiC algorithms and their respective sub-algorithms. This algorithm is divided into two main categories: IDS and ATiC. IDS focuses exclusively on qualitative issues or active problems related to data, specifically Data Change (DC) and Data Theft (DT). These issues are addressed through five sub-algorithms:

- IBD (Irregularity Behavior Detection): This algorithm identifies trusted nodes by detecting anomalies in a system or network based on trust relationships among its components.
- LinBr (Link Break): It prevents link breaks by creating optimal routes with trusted nodes.
- BlaHA (Black Hole Avoidance) and GrHoA (Grey Hole Avoidance): These algorithms deal with attacks where trusted nodes behave as untrusted nodes, such as Black Hole and Grey Hole attacks, respectively. BlaHA checks each reply from neighboring nodes to detect anomalies, while GrHoA focuses on mitigating Grey Hole attacks.
- UnB (Unbelievable Node Avoidance): This algorithm identifies and avoids unreliable nodes by assessing the ratio of received to sent packets, preferring nodes with ratios closer to one.

ATiC focuses on passive quantitative issues and aims to reduce time-related complexity through its five sub-algorithms:

- IGWO (Improved Grey Wolf Optimization Based): This algorithm detects malicious nodes by simulating the social dynamics of a grey wolf pack.
- MulP (Multiple Path towards Destination): It finds multiple routes between the source and destination to improve efficiency.
- PktSiR (Packet Size Regulator): This algorithm adjusts packet size according to the available bandwidth, increasing packet size during low traffic and decreasing it during high traffic.
- MulPKt (Multiple Packet towards Destination): It sends multiple packets simultaneously using the optimal paths determined by MulP.
- MulOR (Multiple Optimal Route between Tx and Rx): This algorithm finds multiple optimal routes between the transmitter and receiver, considering factors like the number of hops and available bandwidth, and then sorts them based on cost before utilizing them for packet transmission.

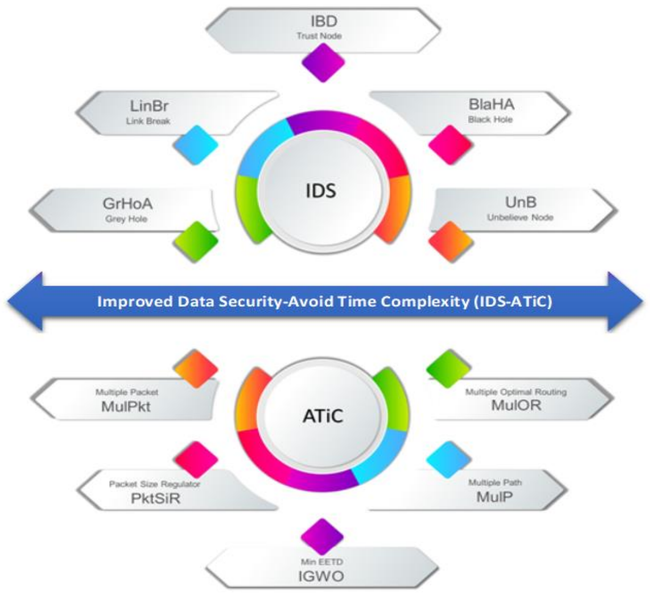


Figure 12 : Improved Data Security – Avoid Time Complexity (IDS-ATiC)

6. Infra-Less KMS dataset

The Infra-Less KMS dataset is a novel compilation tailored for this particular research endeavor. Its attributes are amalgamated from the KDDCup99 and Dataset_Unicauce_V2_87atts datasets, with several attributes being specifically introduced for this study.
Comprising 31 features, only 7 are shared, while the remaining 24 are uniquely utilized to forecast ten distinct types of attacks, namely:

- DoS & DDoS
- Data Change
- Data Theft
- Block Hole
- EETD
- PDR
- (BU) Bandwidth Utilization
- NC (Network Congestion)
- UN (Unbelievable Node)

This dataset was curated from the OmNetpp simulator and the breakdown of its data is detailed in the table below.

Table 1: Display the number of dat allocated for each attack

Total Number of Rows	2,11,010.00
DoS (Denial of Service) & DDoS (Distributed Denial of Service)	32,021.00
DC (Data Change)	10,165.00
DT (Data Theft)	12,010.00
BH (Block Hole)	35,202.00
EETD (End to End Time Delay)	50,151.00
PDR (Packet Delivery Ratio)	42,435.00
BU (Bandwidth Utilization)	9,682.00
NC(Network Congestion)	9,669.00
UN(Unbelievable Node)	9,675.00

Thus, this newly crafted dataset contains ample data for each specified attack outlined in the IDS-ATiC algorithm.

7. Proposed Work

IDS-ATiC is an algorithm that has already been integrated with AODV. In this study, IDS-ATiC is extended by incorporating either Support Vector Machine (SVM) or Gaussian Mixture Model (GMM) as the underlying machine learning (ML) algorithm.

Previous research by the author evaluated ML algorithms and found SVM and GMM to be a suitable choice for implementing IDS-ATiC within the AODV framework, albeit being a supervised learning algorithm requiring labelled attributes. In contrast, unsupervised learning algorithms like GMM can effectively utilize unlabelled attributes. Hence, in this study, both SVM (supervised) and GMM (unsupervised) are being implemented.

The proposed work involves the creation of two separate models: one utilizing SVM and the other utilizing GMM.

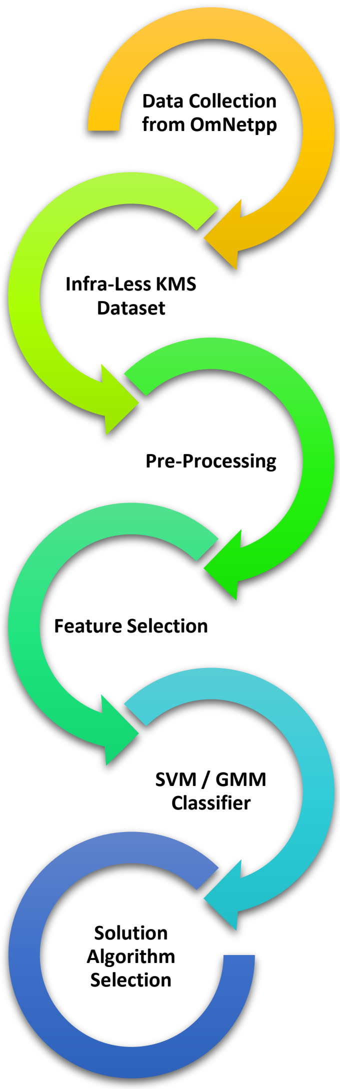


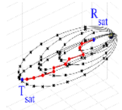
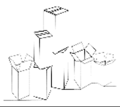
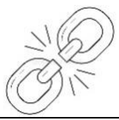
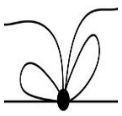
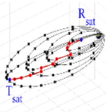
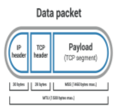
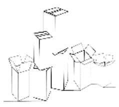
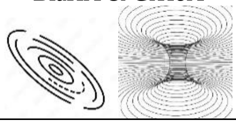
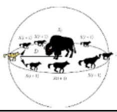
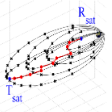
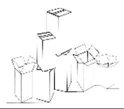
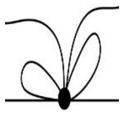
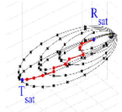
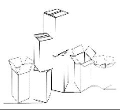
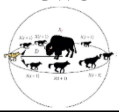
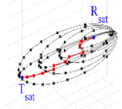
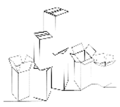
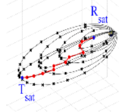
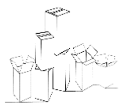
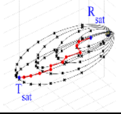
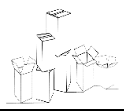
Figure 13: Visualization of the construction process of the SVM

This proposed study aims to identify the most effective machine learning algorithm along with its Problem Finding Time (PFT). The table below illustrates the required combinations of the IDS-ATiC algorithm to address specific scenarios. In cases where a machine learning algorithm is not available, any rule-based problems will be identified. Initial algorithms will then be executed to pinpoint the exact problem scenario. Subsequently, the precise combination of algorithms will be employed to deliver a solution for that particular situation

7.1. Algorithm execution sequence

If the rule-based problem identification is performed during this period, the combined time for situation identification and algorithm execution will range from 119ms to 101ms. The minimum time required for a solution is 101ms, while the maximum time is 119ms. This variance occurs because all initial algorithms must be executed to identify the appropriate situation before the corresponding algorithms can be run, resulting in this duration.

Table 2: Algorithm Execution Sequence

Algorithm No.	**Initial Algorithm**	**Algorithm Sequence**				
1	IBD	MulP	MulOR 	MulPkt 		
Sequence No.	1	2	3	4		
2	LinBr 	UnB 	MulP	MulOR 	PktSiR 	MulPkt 
Sequence No.	1	2	3	4	5	6
2	BlaHA & GrHoA 	IGWO 	MulP	MulOR 	MulPkt 	
Sequence No.	1	2	3	4	5	
4	UnB 	MulP	MulOR 	MulPkt 		
Sequence No.	1	2	3	4		
5	IGWO 	MulP	MulOR 	MulPkt 		
Sequence No.	1	2	3			
6	MulP	MulOR 	MulPkt 			
Sequence No.	1	2	3			
7	MulOR 	MulPkt 				
Sequence No.	1	2				

If a machine learning algorithm is employed and accurately identifies a specific situation, the BlaHA and GrHoA algorithms will require a longer execution time of 64ms each, while the MulOR algorithm will require less execution time, specifically 11ms. These details are derived solely from the author's prior research.

Data collection presents a formidable challenge due to the complexity of identifying and selecting attributes that possess the requisite characteristics for identifying specific problem situations. The author has undertaken this task, deriving certain attributes from existing datasets and creating new ones as required by the algorithms. Detailed explanations regarding the dataset can be found in the dataset section.

7.2. SVM Sample code

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
# Assuming you have your dataset loaded into X (features) and y (target labels)
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize the SVM classifier
svm_classifier = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
# Train the classifier on the training data
svm_classifier.fit(X_train, y_train)
# Make predictions on the testing data
y_pred = svm_classifier.predict(X_test)
# Evaluate the performance of the classifier
print(classification_report(y_test, y_pred))
```

7.3. GMM Sample code

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Fit a separate GMM for each class
gmms = {}
for class_label in set(y_train):
    X_class = X_train[y_train == class_label]
    gmm = GaussianMixture(n_components=5, covariance_type='full')  # Example hyperparameters
    gmm.fit(X_class)
    gmms[class_label] = gmm
# Calculate class probabilities for each data point in the testing set
class_probs = {}
for class_label, gmm in gmms.items():
    class_probs[class_label] = gmm.score_samples(X_test)
# Assign class labels based on the estimated class probabilities
y_pred = [max(class_probs, key=lambda k: class_probs[k][i]) for i in range(len(X_test))]
# Evaluate the performance of the GMM-based classifier
print(classification_report(y_test, y_pred))
```

8. Simulation

OMNeT++ is a versatile, customizable, object-oriented C++ simulation library and framework designed primarily for constructing network simulators. The term 'network' encompasses various types, including wired and wireless communication networks, on-chip networks, and queueing networks. Specific functionalities tailored to domains such as sensor networks, wireless ad-hoc networks, Internet protocols, performance modelling, and photonic networks are provided by independent model frameworks. OMNeT++ features an IDE based on Eclipse, a graphical runtime environment, and a range of additional tools. Extensions are available for real-time simulation, network emulation, database integration, SystemC integration, and various other capabilities.

8.1. Simulation Parameters

Table 3: Simulation Parameter

Parameters	Values
Network Size	1000 X 1000
Number of Nodes	0-500
Max Speed/ Mobility	10 m/s
Pause Time	0-100s
Traffic Model	CBR, VBR
Radio Transmission Range	250m
Routing Protocol	AODV, IDS-ATiC AODV
Simulation Time	600s

Note: This simulation parameters were used to create Inra-Less KMS Dataset

9. Methodology

9.1. Accuracy

Accuracy is calculated by dividing the number of correct predictions by the total number of predictions across all classes. In binary classification, it can be expressed as:

$$\text{Accuracy (ACC)} = (TP + TN) / (TP + TN + FP + FN)$$

9.2. Confusion Matrix and Accuracy

This step is commonly employed in classification methodologies. Here, we assess the accuracy of the trained model and visualize the confusion matrix.

The confusion matrix is a structured table utilized to present the count of accurate and erroneous predictions made by a classification model when the actual values of the test set are available. It follows a specific format.

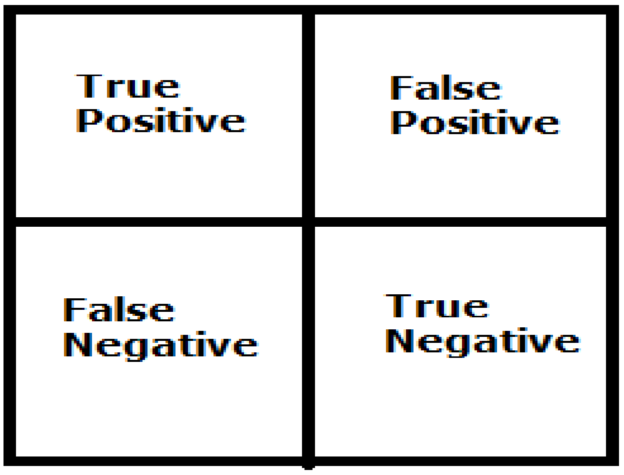


Figure 14: Confusion Matrix for Accuracy

The True values represent the count of accurate predictions. Based on the provided confusion matrix, we deduce that out of 100 test set instances, 90 were accurately classified while 10 were misclassified. Thus, the accuracy of the classification is determined to be 90%.

10. Result and Discussion

The outcomes encompass the prediction accuracy comparison between SVM and GMM, the algorithm execution time (AET) based on predefined rules, the time taken to identify problems (PFT) with SVM and GMM. Lastly, it incorporates the total execution time (TET) for both SVM and GMM.

Table 4: Prediction Accuracy (PA) between SVM and GMM

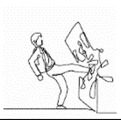
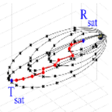
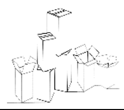
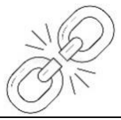
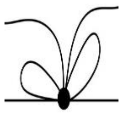
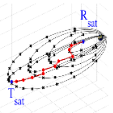
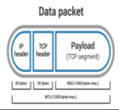
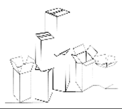
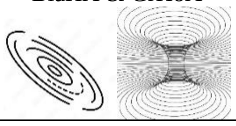
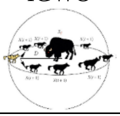
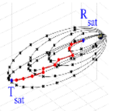
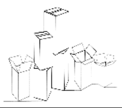
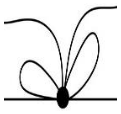
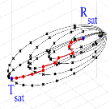
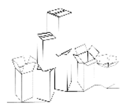
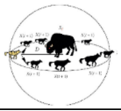
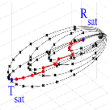
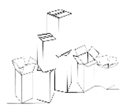
S. No.	Attacks	Accuracy (%)	
		SVM	**GMM**
1	PDR	92	81
2	EETD	78	74
3	DoS	85	77
4	DDoS	85	77
5	DC	71	67
6	DT	94	86
7	BH	83	76
8	BU	83	77
9	NC	80	73
10	UN	86	79

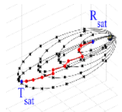
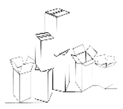
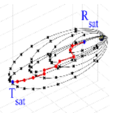
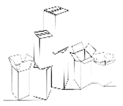
Table 4 illustrates the Prediction Accuracy (PA) comparison between SVM and GMM. SVM exhibits lower PA compared to GMM. Both algorithms operate within the realm of multiple class classification. Notably, higher PA correlates with better results in terms of Time Complexity and Security Complexity. The PA is computed based on 31 attributes and 2,11,010 samples

In Table 5, each row presents a set of algorithms employed for execution under specific circumstances. The table displays the time required for each algorithm to run within the given simulation parameters and defined scenarios. The study encompasses two distinct scenarios: one without intruders and the other with 15% intrusion. The computation of algorithm execution time is conducted within the context of the latter scenario.

Following table showing Algorithm Execution Time (AET) of each algorithms sequence

Table 5: Algorithm Execution Time (AET)

Tot. Exec. Time	**Initial Algorithm**	**Algorithm Sequence**				
36ms	IBD 	MulP	MulOR 	MulPkt 		
Execution Time	**20ms**	**5ms**	**10ms**	**1ms**		
51ms	LinBr 	UnB 	MulP	MulOR 	PktSiR 	MulPkt 
Execution Time	**15ms**	**2ms**	**5ms**	**10ms**	**18ms**	**1ms**
64ms	BlaHA & GrHoA 	IGWO 	MulP	MulOR 	MulPkt 	
Execution Time	**25ms**	**23ms**	**5ms**	**10ms**	**1ms**	
18ms	UnB 	MulP	MulOR 	MulPkt 		
Execution Time	**2ms**	**5ms**	**10ms**	**1ms**		
39ms	IGWO 	MulP	MulOR 	MulPkt 		
Execution Time	**23ms**	**5ms**	**10ms**	**1ms**		
16ms	MulP	MulOR	MulPkt			

			
Execution Time	**5ms**	**10ms**	**1ms**
11	MulOR 	MulPkt 	
Execution Time	**10ms**	**1ms**	

According to these findings, the highest execution time recorded is 64ms, observed with the BlaHA and GrHoA algorithms. Conversely, the lowest execution time among the algorithms is 11ms.

Problem Finding Time (PFT)

Table 6: Problem Finding Time (PFT) between SVM and GMM

S. No.	Attacks	Time (ms)	
		SVM	GMM
1	PDR	1.4	1.62
2	EETD	1.73	1.59
3	DoS	3.26	4.01
4	DDoS	3.72	4.44
5	DC	6.39	5.69
6	DT	6.13	5.53
7	BH	6.02	5.76
8	BU	1.58	2.09
9	NC	1.16	1.69
10	UN	1.05	1.18

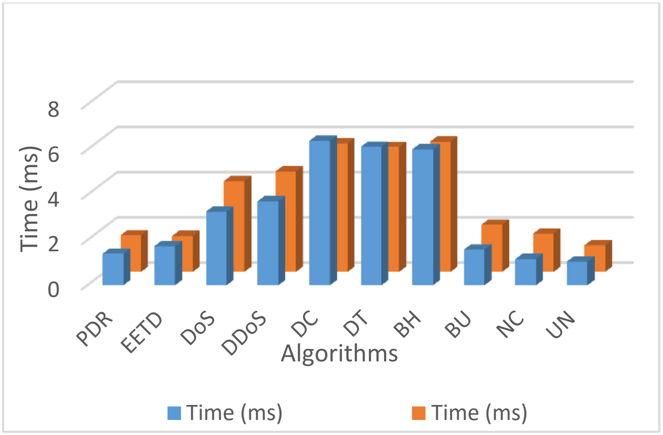


Figure 15: Comparison of Problem Finding Time (PFT)

The identification of Problem Finding Time is essentially a forecasting task facilitated by Machine Learning algorithms. This study employed two algorithms, namely SVM and GMM, operating on the Infra-Less KMS dataset comprising 2,11,010 samples and 31 attributes. The findings indicate that GMM demonstrates a shorter qualitative Problem Finding Time, suggesting it requires less time to predict qualitative issues compared to SVM.

While GMM excels in this aspect, SVM exhibits a generally satisfactory performance in Problem Finding Time. The graph below illustrates the comparison of PFT durations between SVM and GMM. Ultimately, SVM demonstrates a moderately good overall performance in Problem Finding Time.

Total Execution Time (TET)

Table 7: Total Execution Time (TET) with SVM

S. No.	Attacks	Time (ms)		
		PFT	AET	TET
1	PDR	1.4	39	40.4
2	EETD	1.73	51	52.73
3	DoS	3.26	64	67.26
4	DDoS	3.72	69.23	70.39
5	DC	6.39	64	70.39
6	DT	6.13	64	70.13
7	BH	6.02	64	70.02
8	BU	1.58	29	30.58
9	NC	1.16	29	30.16
10	UN	1.05	39	40.05

Table 7 presents the Total Execution Time (TET) of the SVM. TET is computed by adding two durations: Problem Finding Time (PFT) and Algorithm Execution time. PFT is solely determined by the ML algorithm used. According to the findings, both algorithms yield identical TET values except in the case of DDoS, where SVM demonstrates superior TET performance.

Figure 16 illustrates the comparison of Total Execution Times (TETs) between SVM and GMM. The results suggest that both algorithms are suitable for implementation in IDS-ATiC AODV. Each algorithm is deemed appropriate for specific situations, contributing to a reduction in prediction time.

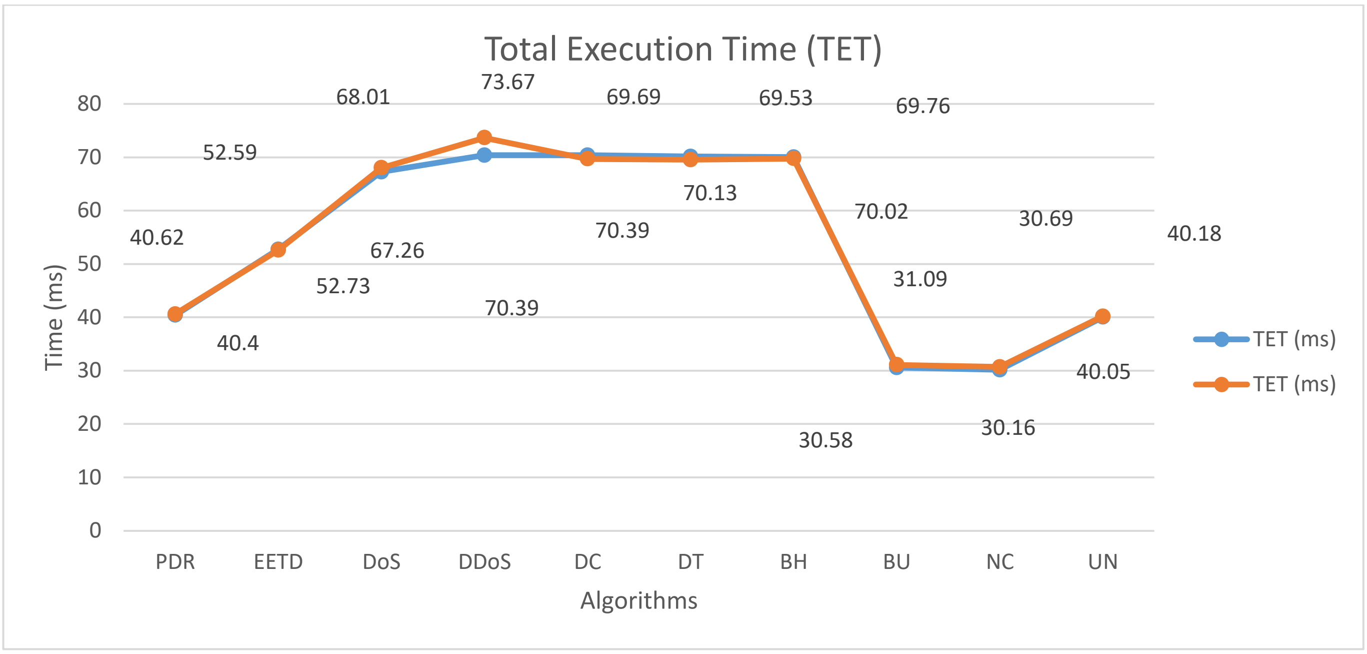


Figure 16: Comparison of Total Execution Time (TET) between SVM and GMM

Table 8: Total Execution Time (TET) with GMM

S. No.	Attacks	Time (ms)		
		PFT	AET	TET
1	PDR	1.62	39	40.62
2	EETD	1.59	51	52.59
3	DoS	4.01	64	68.01
4	DDoS	4.44	69.23	73.67
5	DC	5.69	64	69.69
6	DT	5.53	64	69.53
7	BH	5.76	64	69.76
8	BU	2.09	29	31.09
9	NC	1.69	29	30.69
10	UN	1.18	39	40.18

11. Conclusion

This study assessed the implementation of two algorithms, SVM and GMM, within the context of IDC-ATiC AODV. Traditionally, solution algorithms were selected based on rule-based approaches. However, in this study, the time taken for execution varied from 119ms to 101ms when employing machine learning (ML) algorithms to predict problems. This time was evaluated using Problem Finding Time (PFT) and Algorithm Execution Time (AET).

In the case of SVM, the Total Execution Time (TET) ranged from 70.39ms to 30.16ms. SVM required only 59.15% of the time for executing the solution algorithm during higher time periods, resulting in a reduction of approximately 40.85% compared to previous execution times. Conversely, during lower time periods, SVM only utilized 29.86% of the time compared to previous methods, leading to a reduction of about 70.14%.

Similarly, for GMM, the TET ranged from 73.67ms to 30.69ms. GMM utilized 61.90% of the time during higher time periods, reducing execution time by approximately 38.10%. During lower time periods, GMM utilized only 30.38% of the time compared to previous methods, resulting in a reduction of about 69.62%.

Overall, SVM emerged as the more favorable ML algorithm for IDS-ATiC AODV, offering greater time efficiency.

Future endeavors should focus on improving the effectiveness of multiple class classification within the implementation. Additionally, deployment aspects were not addressed in this study, indicating a need for their inclusion in future work.

References

[1] N. Kanimozhi, S. Hari Ganesh 2, B. Karthikeyan, "An Analysis of Machine Learning Solution for QoS and QoE in Network (Infrastructure Oriented and Less)", International Journal of Computer Sciences and Engineering, May, Vol.11, Issue.5, pp.41-59, 2023. ISSN: 2347-2693 (Online)

[2] N. Kanimozhi, S. Hari Ganesh 2, B. Karthikeyan, “Performance Analysis of MANET Routing Protocols”, International Journal of Computer Applications (0975 – 8887) December, Vol.185, No. 50, pp-44-50, 2023.

[3] N. Kanimozhi, S. Hari Ganesh 2, B. Karthikeyan, "Irregularity Behaviour Detection - Ad-hoc On-Demand Distance Vector Routing Protocol (IBD - AODV): A Novel Method for Determining Unusual Behaviour in Mobile Ad-hoc Networks (MANET)", International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169 September, Vol.11 Issue: 9, pp.1098-1010, 2023.

[4] N. Kanimozhi, S. Hari Ganesh 2, B. Karthikeyan, "Minimizing End-To-End Time Delay in Mobile Ad-Hoc Network using Improved Grey Wolf Optimization Based Ad-Hoc On-demand Distance Vector Protocol (IGWO-AODV)", International Journal on Recent and Innovation Trends in Computing and

Communication ISSN: 2321-8169, September, Vol.11, Issue:9, pp.1111-1115, 2023.

[5] B.Karthikeyan, N. Kanimozhi and Dr.S.Hari Ganesh, “Analysis of Reactive AODV Routing Protocol for MANET”, IEEE Xplore (978-1-4799-2876-7), Oct, pp. 264-267, 2014.

[6] B.Karthikeyan, N. Kanimozhi and Dr.S.Hari Ganesh, “Security and Time Complexity in AODV Routing Protocol”, International Journal of Applied Engineering Research (ISSN:0973-4562), Vol. 10, No.20,June 2015, pp.15542- 155546. – Scopus Indexed.

[7] B. Karthikeyan, Dr.S.Hari Ganesh and N. Kanimozhi, “Encrypt - Security Improved Ad Hoc On Demand Distance Vector Routing Protocol (En-SIm AODV)”, ARPN Journal of Engineering and Applied Sciences (ISSN: 1819-6608), Vol. 11, No. 2, January, pp.1092-1096, 2016.

[8] B. Karthikeyan,Dr.S.Hari Ganesh and Dr. JG.R. Sathiaseelan, “ Optimal Time Bound Ad-Hoc On-demand Distance Vector Routing Protocol (OpTiB-AODV)”, International Journal of Computer Applications (ISSN:0975 – 8887), April, Vol.140, No.6, pp 7-11, 2016.

[9] B. Karthikeyan, Dr.S.Hari Ganesh, Dr. JG.R. Sathiaseelan and N. Kanimozhi , “High Level Security with Optimal Time Bound Ad-Hoc On-demand Distance Vector Routing Protocol (HiLeSec-OpTiB AODV)”,International Journal of Computer Science Engineering(E-ISSN: 2347-2693), April, Vol.4, No.4, pp.156-164, 2016.

[10] Jhansi Rani Kaka and K. Satya Prasad, “Differential Evolution and Multiclass Support Vector Machine for Alzheimer’s Classification”, Hindawi, Security and Communication Networks, Vol.2022, Article ID 7275433, 13 pages.

[11] Khalid Abualsaud, Elias Yaacoub, Maazen Alsabaan and Mohsen Guizani, “Lightweight Multi-Class Support Vector Machine-Based Medical Diagnosis System with Privacy Preservation” Sensors, 23, 9033, 2023. https:// doi.org/10.3390/s23229033.

[12] Ceren Atik, Recep Alp Kut, Reyat Yilmaz, and Derya Birant, "Support Vector Machine Chains with a Novel Tournament Voting", Electronics, 12, 2485, 2023. https://doi.org/10.3390/electronics12112485.

[13] Zhi Quan and Luoxi Pu, "An improved accurate classification method for online education resources based on support vector machine(SVM): Algorithm and experiment" Education and Information Technologies, 28: pp.8097–8111, 2023.

[14] Siva Rajesh Kasa & Vaibhav Rajan, "Avoiding inferior clusterings with misspecified Gaussian mixture models", 2 Vol:.(1234567890)Scientific Reports | (2023) 13:19164 | https://doi.org/10.1038/s41598-023-44608-3 www.nature.com/scientificreports/

[15] Abdullahi Abubakar Mas’ud, Arunachalam Sundaram, Jorge Alfredo Ardila-Rey, Roger Schurch, Firdaus Muhammad-Sukki and Nurul Aini Bani, "Application of the Gaussian Mixture Model to Classify Stages of Electrical Tree Growth in Epoxy Resin", Sensors 2021, 21, 2562. https://doi.org/10.3390/s21072562